HOW TO ENABLE HPC SYSTEM DEMAND RESPONSE: AN EXPERIMENTAL STUDY

Kishwar Ahmed
Florida International University

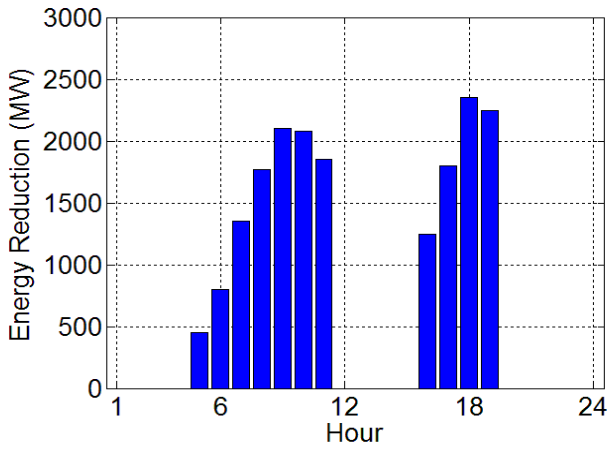
Kazutomo Yoshii (speaker)
Argonne National Laboratory

Outline

- Motivation
- DVFS-based Demand Response
- Power-capping-based Demand Response
- Experiments on Chameleon Cluster
- Conclusions

What is Demand Response (DR)?

- DR: Participants reduce energy consumption
  - During transient surge in power demand
  - Other emergency events
- A DR example:
  - Extreme cold in beginning of January 2014
  - Closure of electricity grid
  - Emergency demand response in PJM and ERCOT



Energy reduction target at PJM on January 2014

Demand Response Is Popular!

SUSTAINABILITY

Why Apple Is Getting into the Energy Business

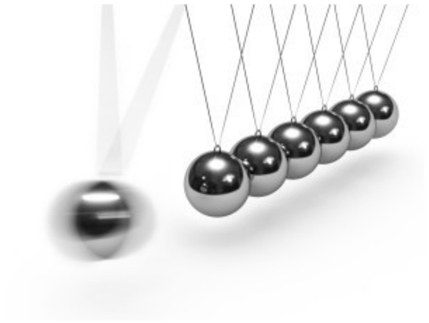
by Peter Fox-Penner

NOVEMBER 25, 2016

But solar electricity is only the beginning of the future energy marketplace. Many companies are already in markets where "demand-response" contracts enable them to sell the right to manage a portion of their power use, allowing them to be paid for reducing their energy during hours when the spot price of power is high. In the future, in addition to selling actual

Equinix 'in R&D phase' of demand response experiments

OCTOBER 16, 2015 BY BRENDAN COYNE — 1 COMMENT



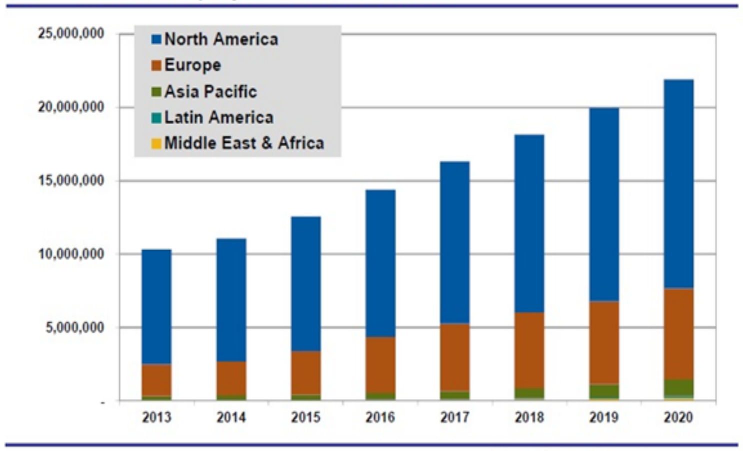
Equinix is "in the R&D phase" of testing demand response technologies, according to UK MD Russell Poole.

National Grid wants more businesses to help balance the electricity system by turning power off or on, or adjusting loads, in return for payment. However, mission critical sites have traditionally been reluctant to increase any perceived risk factors for relatively low rewards. National Grid though, is rethinking its mechanisms and contract structures.

DEMAND RESPONSE WILL DOUBLE BY 2020: HERE'S WHY

BY JESSICA KENNEDY SEPTEMBER 3, 2014

Chart 1.1 Total DR Sites by Region, World Markets: 2013-2020



(Source: Navigant Research)

HPC System as DR Participant?

- HPC system is a major energy consumer
  - China’s 34-petaflop Tianhe-2 consumes 18MWs of power
    - Can supply small town of 20,000 homes
  - The power usage of future HPC system is projected to increase
    - Future exascale supercomputer has power capping limit
    - But not possible with current system architecture
- Demand response aware job scheduling envisioned as possible future direction by national laboratories [“Intelligent Job Scheduling” by Gregory A. Koenig-ORNL]

HPC System as DR Participant? (Contd.)

- A number of recent surveys on possibility of supercomputer's participation in DR program
- Patki et al. (in 2016)
  - A survey to investigate demand response participation of 11 supercomputing sites in US
  - "*…SCs in the United States were interested in a tighter integration with their ESPs to improve Demand Management (DM).*"
- Bates et al. (in 2015)
  - "*…the most straightforward ways that SCs can begin the process of developing a DR capability is by enhancing existing system software (e.g., job scheduler, resource manager)*"

Power-capping

- What is power-capping?
  - Control knobs that allow users to specify the upper limit of power consumption of CPUs, memory or the entire node
- Power-capping is important
  - To achieve global power cap for the cluster
- Power-capping is common in modern processors
  - Intel processors support power capping through RAPL interface
    - opportunistically adjusts voltage and frequency based on thermal and energy constraints
  - AMD processors' Advanced Power Management Link (APML) technology
  - NVIDIA GPU's NVIDIA Management Library (NVML)

Related Works

- Data center and smart building demand response
  - Workload scheduling: such as load shifting in time, geographical load balancing
  - Resource management: server consolidation, speed-scaling
- However,
  - These approaches are applicable for internet transaction-based data center workload
  - Service time for data center workload are assumed uniform and delay-intolerant
- HPC system demand response
  - Recently, we are proposing HPC system demand response model
    - Based on
      - dynamic voltage frequency scaling (DVFS)
      - Power capping

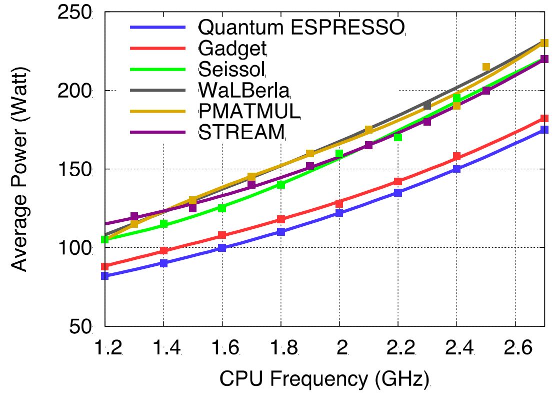
DVFS-based Demand Response

DVFS-based Demand Response

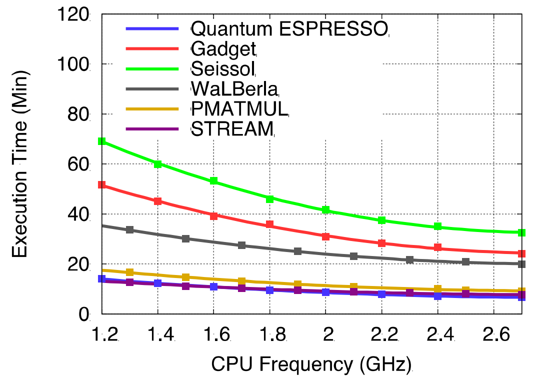
- Power and performance prediction model
  - Based on a polynomial regression model
- Resource provisioning
  - Determine processors’ optimal frequency to run the job
- Job scheduling
  - Based on FCFS with possible job eviction (to ensure power bound constraint)

Power and Performance Prediction

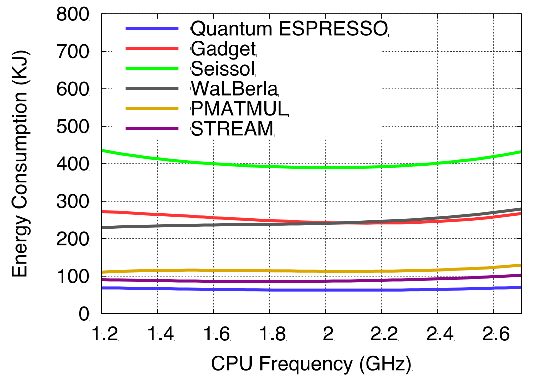
$$p(j,f) = a + b \cdot f + c \cdot f^2 + d \cdot f^3$$



$$t(j,f) = \alpha + \beta \cdot f + \gamma \cdot f^2$$



$$e(j,f) = n_j \cdot p(j,f) \cdot t(j,f)$$



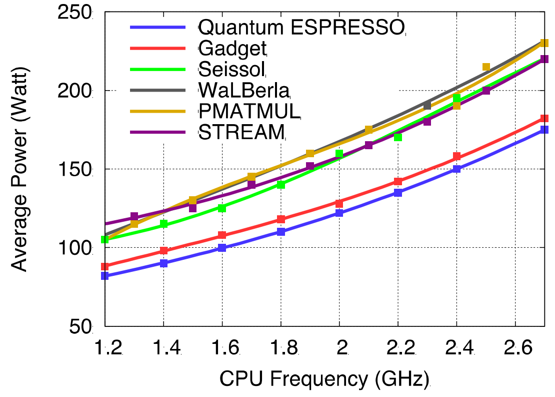
Power and Performance Prediction

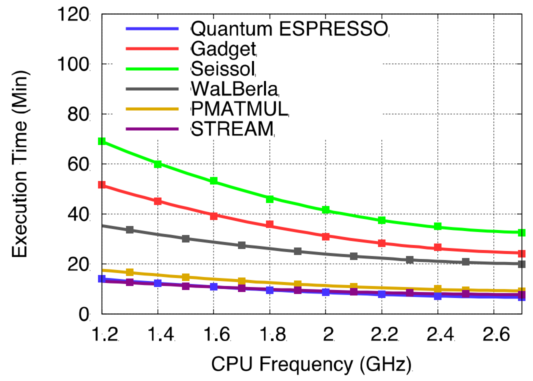
$$p(j,f) = a + b \cdot f + c \cdot f^2 + d \cdot f^3$$

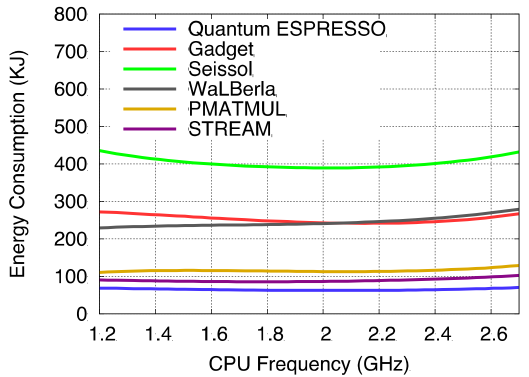
P = A*C*V^2*f + Pstatic
V is proportional to f
=> P ≈ f^3

$$t(j,f) = \alpha + \beta \cdot f + \gamma \cdot f^2$$

$$e(j,f) = n_j \cdot p(j,f) \cdot t(j,f)$$







Optimal Frequency Allocation

- Determine optimal frequency such that
  - Energy consumption is optimized during demand response period
  - Highest frequency during normal periods to ensure highest performance

Minimize: $\sum_{j \in R} e_R(j, f_j)$

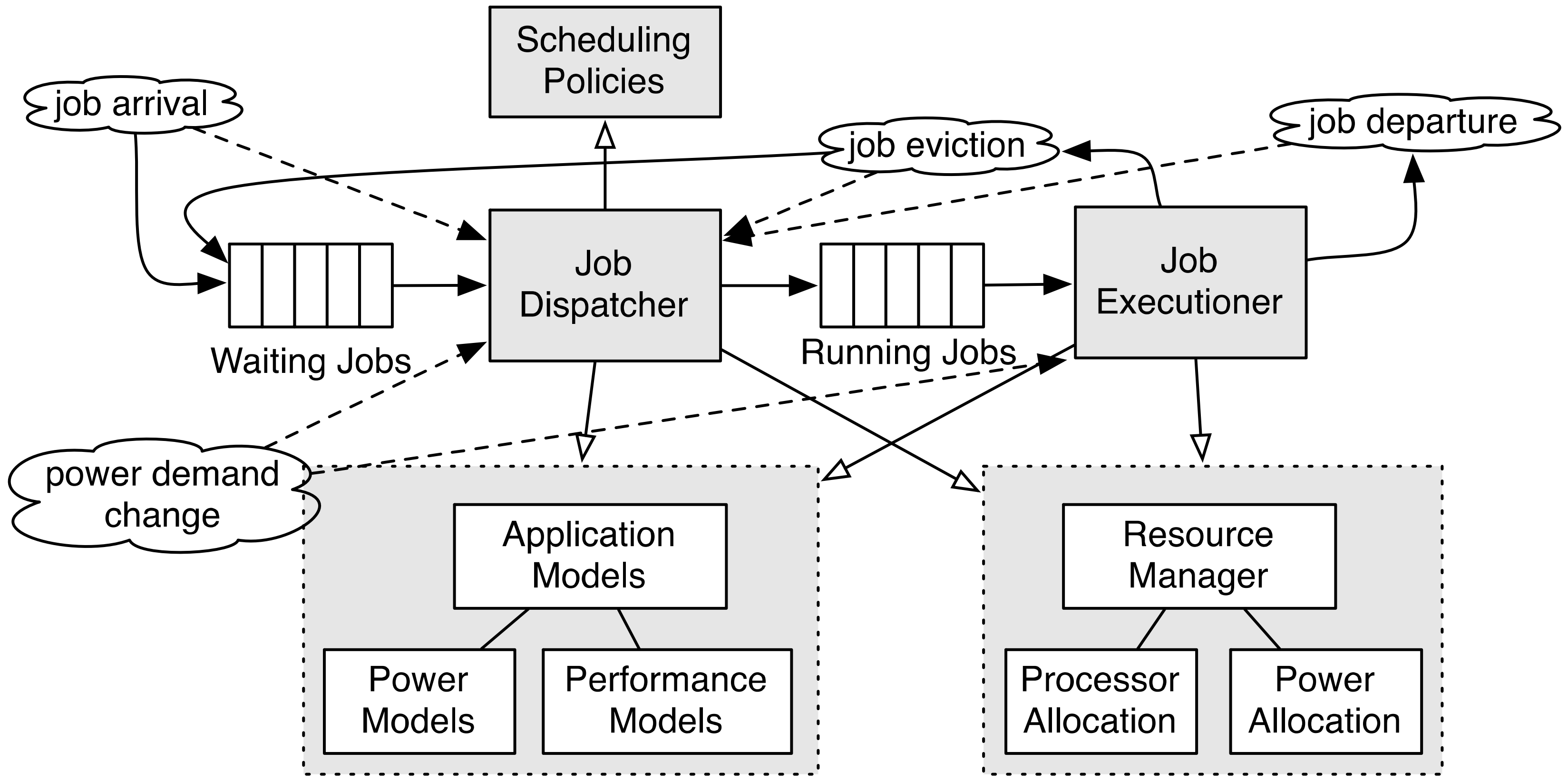
subject to constraints (4) and (5)

$$e_R(j, f_j) = (1 - \alpha_j) \cdot n_j \cdot p(j, f_j) \cdot t(j, f_j)$$

$$f_{min} \leq f_j \leq f_{max} \quad (4)$$

$$p_{run} = \sum_{j \in R} p(j, f_j) \leq \hat{p} \quad (5)$$

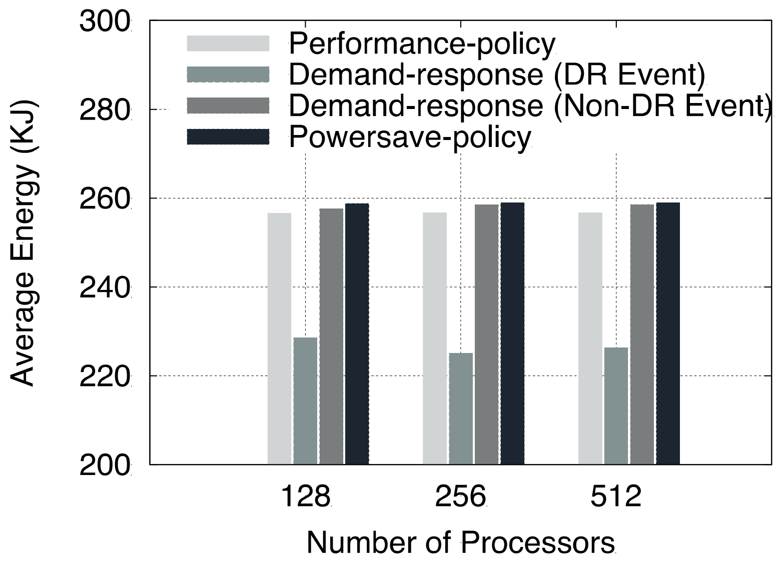
Job Scheduler Simulator (Contd.)

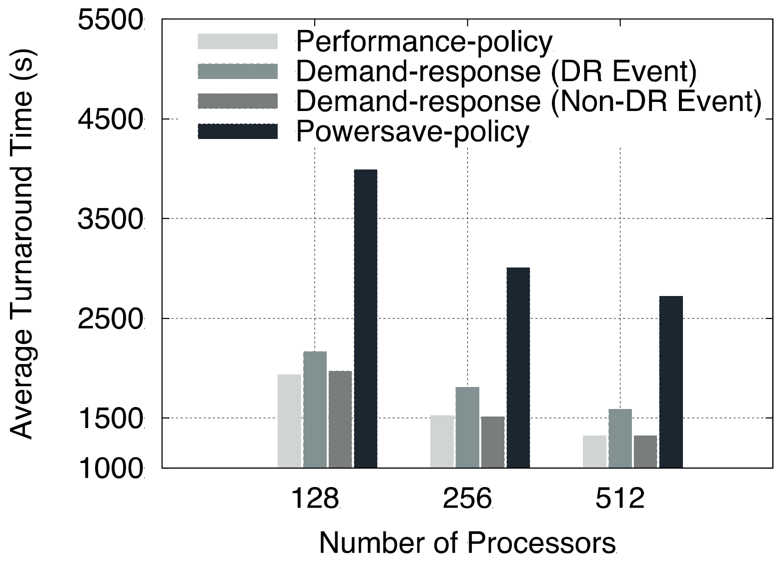


Experiment

- Workload trace collected from Parallel Workloads Archive
- Power and performance data collected from literature for HPC applications
- Two scheduling policies
  - Used in Linux kernel of Intel processors
  - Performance-policy
    - Always chooses maximum frequency to ensure best application runtime
  - Powersave-policy
    - Always chooses the minimum frequency to minimize the power consumption

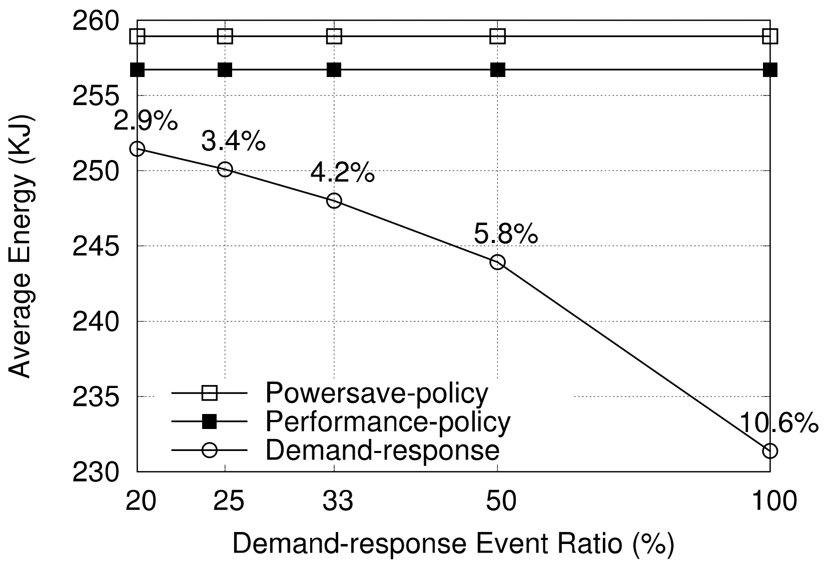
Energy vs. Performance

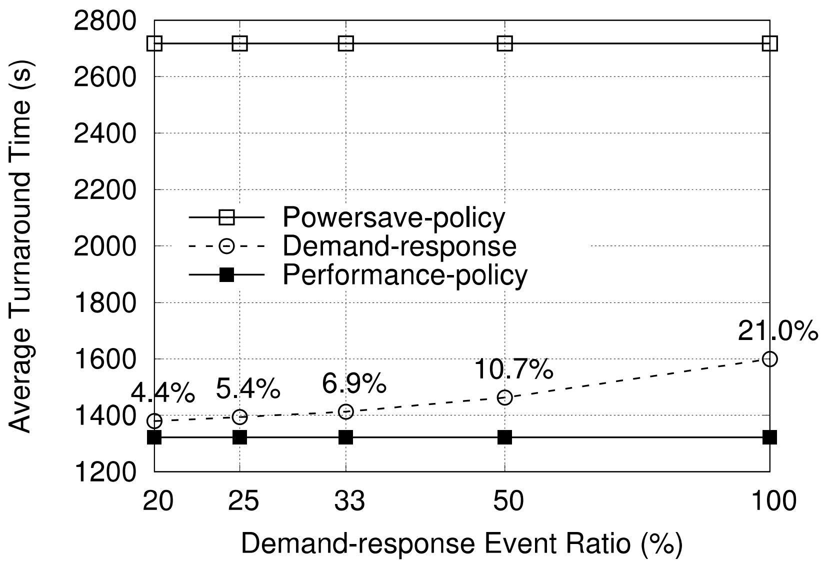




Observation: Reduced energy consumption with focus on demand response periods

Impact of Demand-response Event Ratio





Observation: Average energy decreases with longer demand response event

Power-capping-based Demand Response

Applications and Benchmarks

Benchmark Type	Description	Applications	Application Description
Scalable science benchmarks	Expected to run at **full scale** of the CORAL systems	HACC, **Nekbone**, etc.	Compute intensity, small messages, allreduce
Throughput benchmarks	Represent large ensemble runs	UMT2013, AMG2013, SNAP **LULESH**, etc.	Shock hydrodynamics for unstructured meshes.
Data Centric Benchmarks	Represent emerging **data intensive** workloads – Integer operations, instruction throughput, indirect addressing	Graph500, **Hash,** etc.	Parallel hash benchmark
Skeleton Benchmarks	Investigate various **platform characteristics** including network performance, threading overheads, etc.	CLOMP, **XSBench,** etc.	Stresses system through memory capacity.

Applications and Benchmarks (Contd.)

Benchmark Type	Description	Applications	Application Description
NAS Parallel Benchmarks	A small set of programs designed to help evaluate the **performance of parallel supercomputers**	IS, EP, FT, **CG**	CG - Conjugate Gradient method
Dense-matrix multiply benchmarks	A simple, multi-threaded, dense-matrix multiply benchmark. The code is designed to measure the sustained, floating-point computational rate of a single node	MT-DGEMM, Intel MKL DGEMM	MT-DGEMM: The source code given by NERSC (National Energy Research Scientific Computing Center) Intel MKL DGEMM: The source code given by Intel to multiply matrix
Processor Stress Test Utility	N/A	FIRESTARTER	Maximizes the energy consumption of 64-Bit x86 processors by generating heavy load on the execution units as well as transferring data between the cores and multiple levels of the memory hierarchy.

Measurement Tools

- etrace2
  - Reports energy and execution time of an application
  - Relies on the Intel RAPL interface
  - Developed under DOE COOLR/ARGO project

- An example run

```
../tools/pycoolr/clr_rapl.py --limitp=140
etrace2 mpirun -n 32 bin/cg.D.32

../tools/pycoolr/clr_rapl.py --limitp=120
etrace2 mpirun -n 32 bin/cg.D.32
```

```
Output:
p0 140.0
p1 140.0


 NAS Parallel Benchmarks 3.3 -- CG Benchmark


 Size:    1500000
 Iterations:   100
 Number of active processes:    32
 Number of nonzeroes per row:       21
 Eigenvalue shift: .500E+03


   iteration           ||r||                 zeta
        1       0.73652606305295E-12    499.9996989885352
...
# ETRACE2_VERSION=0.1
# ELAPSED=1652.960293
# ENERGY=91937.964940
# ENERGY_SOCKET0=21333.227051
# ENERGY_DRAM0=30015.779454
```

Measurement Tools (Contd.)

- pycoolr
  - Measure processor power usage and processor temperature
  - Use Intel RAPL capability to measure power usage
  - Power capping limit change capability
  - Reports data in json format
- An example run

```
../tools/pycoolr/clr_rapl.py --limitp=140
mpirun -n 32 ./nekbone ex1

./coolrs.py > nekbone.out
```

{"sample":"**temp**","time":1499822397.016,"node":"protos","p0":{**"mean":34.89** ,"std":1.20 ,"min":33.00 ,"max":36.00 ,"0":33,"1":33,"2":35,"3":36,"4":35,"5":36,"6":36,"7":34,"pkg":36}}

{"sample":"**energy**","time":1499822397.017,"node":"protos","label":"run","energy":{"p0":57706365709,"p0/core":4262338717,"p0/dram":62433931283,"p1":15467688771,"p1/core":18329000806,"p1/dram":55726072673},"power":{**"p0":16.3**,"p0/core":4.6,"p0/dram":1.4,**"p1":16.7**,"p1/core":4.8,"p1/dram":0.9,"total":35.3},"powercap":{"p0":140.0,"p0/core":0.0,"p0/dram":0.0,"p1":140.0,"p1/core":0.0,"p1/dram":0.0}}

Experimental Testbed

- Experimental node@Tinkerlab
  - Intel Sandy Bridge processor
  - Provide power-capping capability
  - Consists of 2 processors with 32 cores

Power and Performance Prediction

- We use third-order polynomial function to determine power usage of job *j* running at processors' power-cap limit p_c:

$$p(j, p_c) = a + b \cdot p_c + c \cdot {p_c}^2 + d \cdot {p_c}^3$$

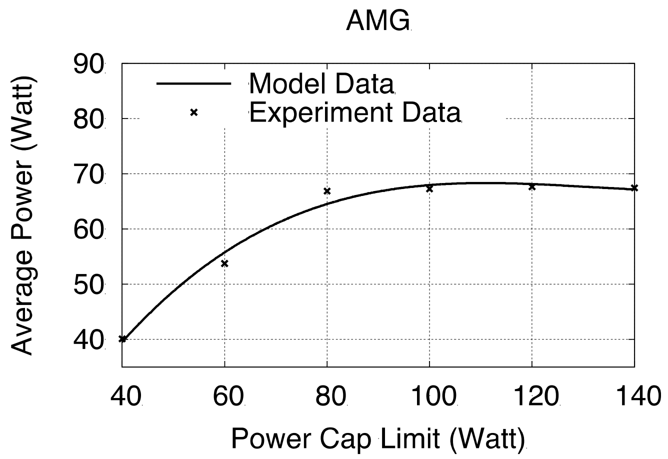
- We use exponential regression function to determine execution time:

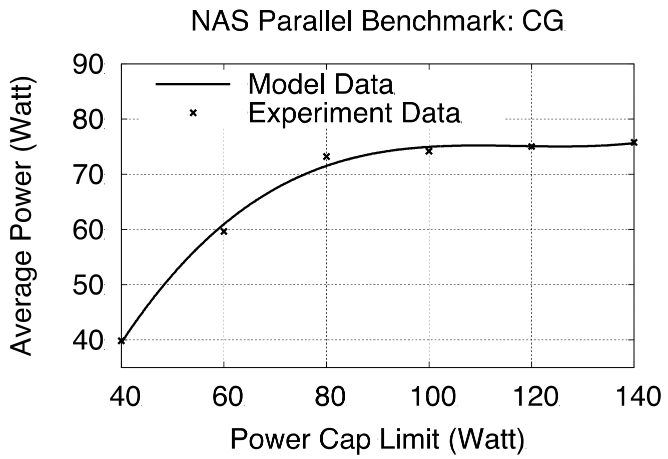
$$t(j, p_c) = \alpha \cdot e^{\beta \cdot p_c} + \gamma$$

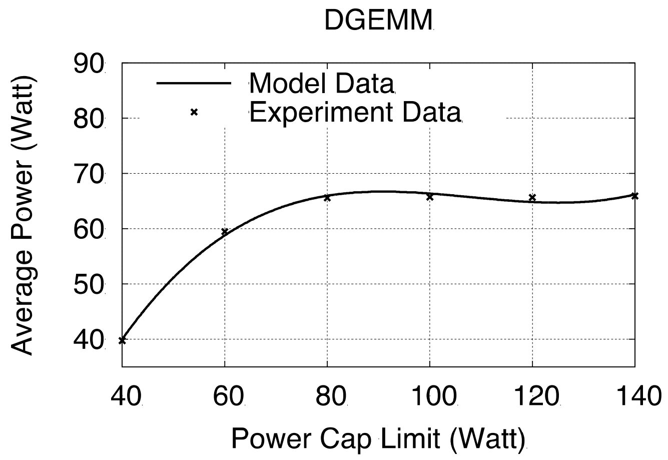
- Total energy consumption for job *j* can be determined as following:

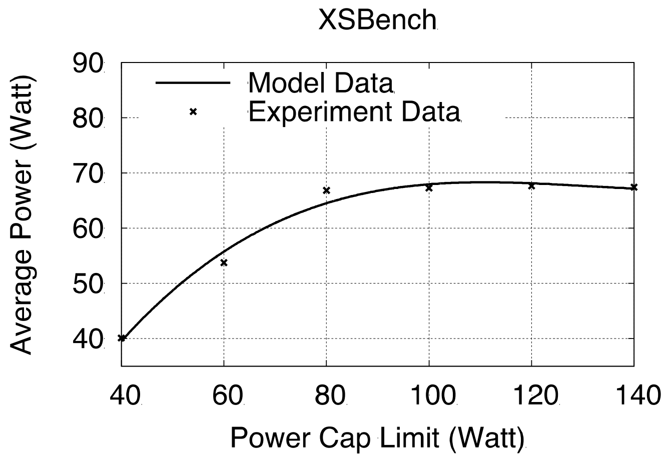
$$e(j, p_c) = n_j \cdot p(j, p_c) \cdot t(j, p_c)$$

Power and Performance Prediction Results

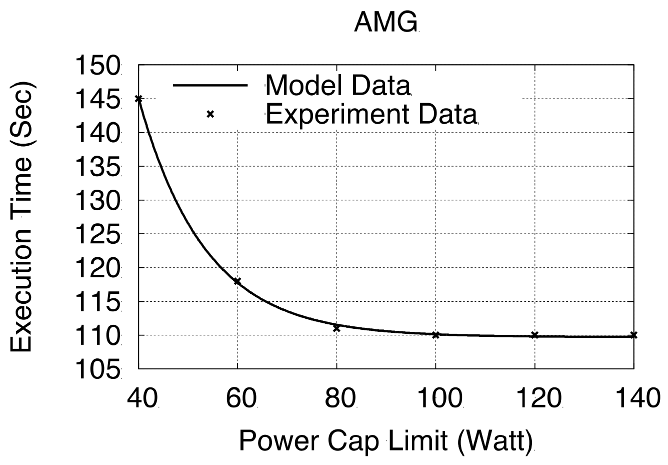


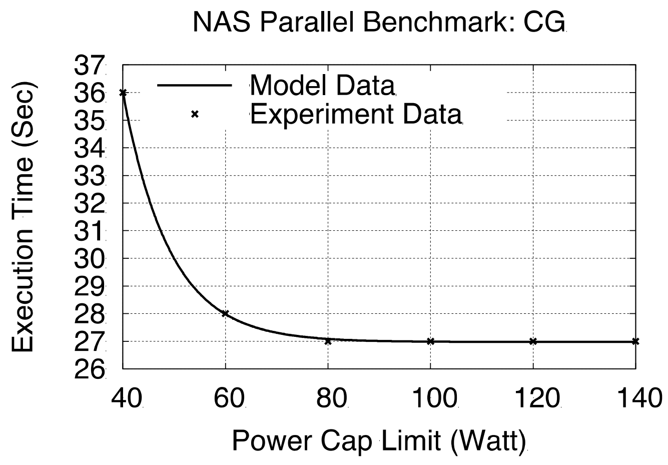


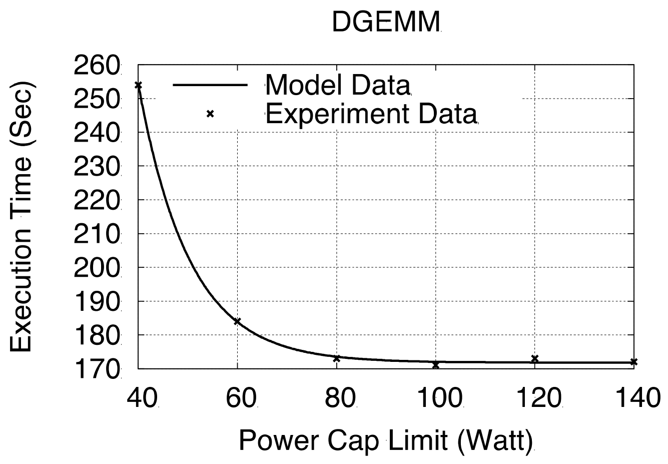


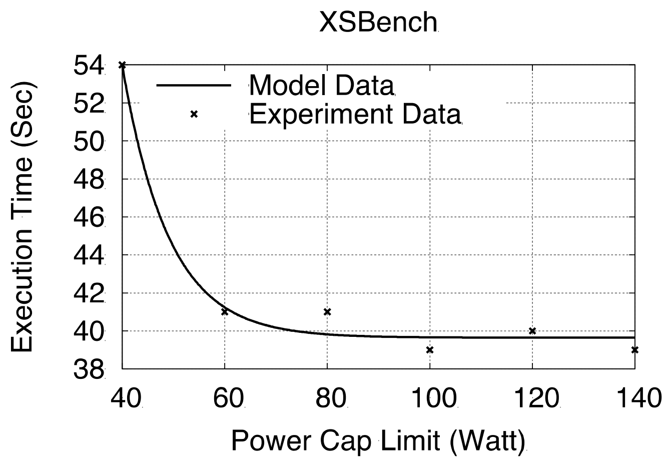


Power and Performance Prediction Results (Contd.)







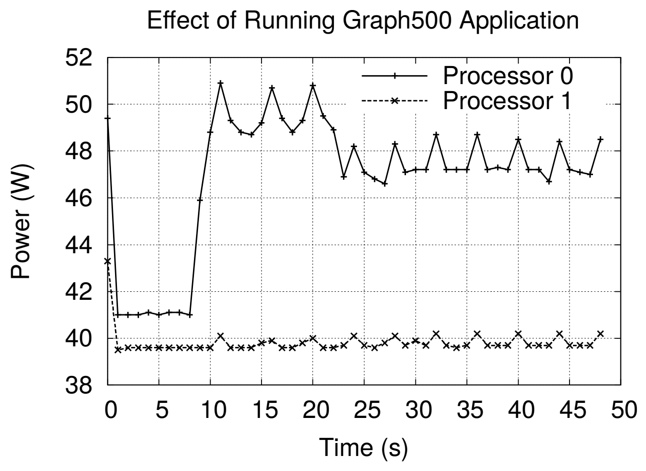


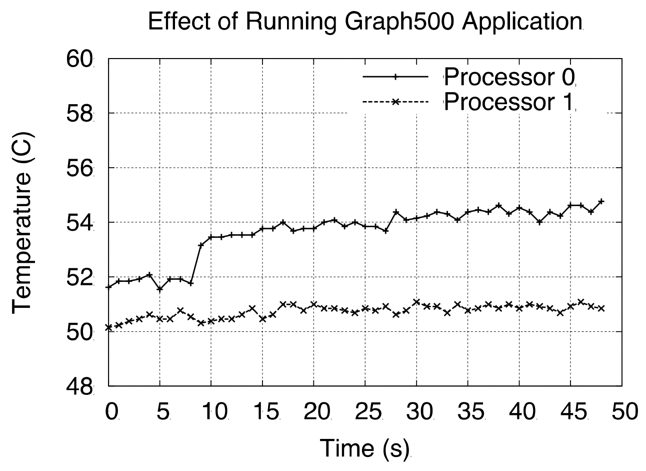
Experiments at Chameleon Cluster

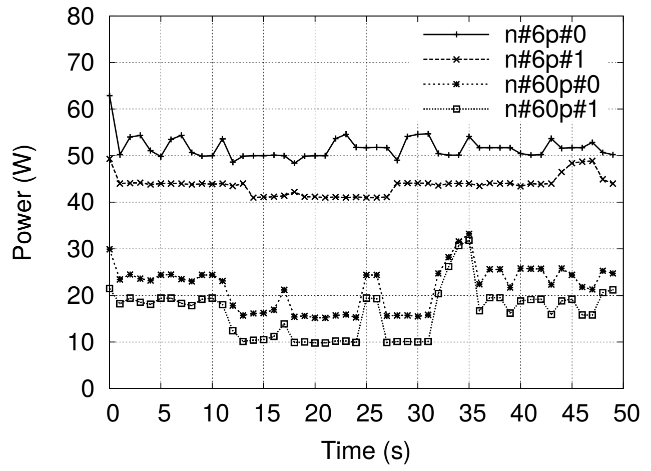
Experiments at Chameleon Cluster

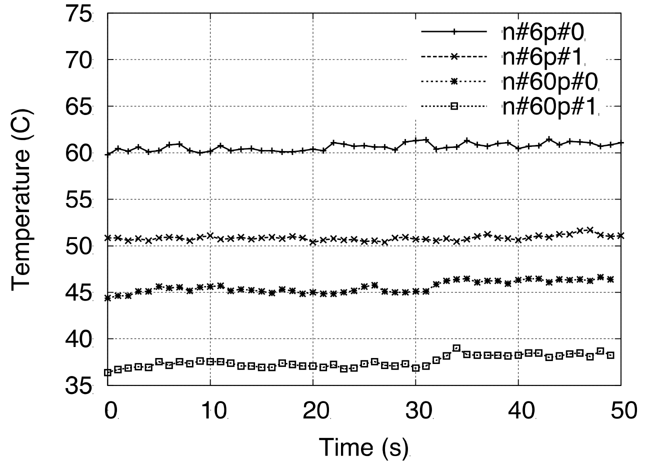
- We want to –
  - Show demand response participation model can be feasible in real-life setup
    - Use Chameleon cluster for such experiments
  - Measure power and performance
    - Using tools such as pycoolr, etrace2, and racadm
  - Run MPI-based applications
    - Using multiple nodes inside Chameleon cluster
  - Implement a scheduler algorithm inside the Chameleon
    - To show effectiveness of demand response model

Application Execution@Chameleon









Power-capping Inside Chameleon

- We initially tried to use pycoolr tool to cap power
  - But faced some difficulties with RAPL availability on DELL servers at Chameleon
    - DELL BIOS locked RAPL power cap
- We have been using Dell RACADM tool
  - To measure power usage at runtime
  - To cap power at different limit

Applications on Multiple Nodes

- Running MPI-based applications using existing complex appliances on MPI protocol
- Based on the runs, we scale to large number of nodes
  - Adaptive Energy and Power Consumption Prediction (AEPCP) model for prediction to large node number
- Use the experiment results to enable demand response
  - Exploiting variation in number of nodes per job
  - Exploiting power capping property

Conclusions

- We studied
  - Possibility of HPC system’s demand response participation
- We proposed a demand-response model which ensures
    - Demand response participation through frequency variation, power capping and processor allocation
- We experimented
  - Real-life scientific applications on experiment cluster
  - Demonstrated effectiveness of our proposed approaches
- Goal
  - Running applications on multiple nodes with power-capping property
  - Show effectiveness of demand response participation on real cluster modifying scheduling algorithm

Thank you all! Questions?